



# **ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**

**Кафедра  
«Криптология и кибербезопасность»**

---

## **Лабораторная работа №2**

по предмету «Технологии контейнеризации»

Выполнил студент группы Б20-505

**Сорочан Илья**

---

**Москва – 2023**

---

# Содержание

1. Vagrant disk .....	7
1.1. Подготовительные работы .....	7
1.2. Работа с дисками .....	7
2. Vagrant network .....	9
2.1. Vagrantfile .....	9
2.2. Проверка всех измененных параметров .....	10
3. Vagrant provision .....	12
3.1. Установка .....	12
3.2. Подтверждение установки .....	12
4. Vagrant multi-machine .....	14

## Предисловие

Перед началом хода лабораторной работы стоит сказать несколько слов про её предыдущую версию. Дело в том, что вся работа уже была выполнена за исключением одной небольшой детали – использовались статичные айпишники вида 192.168.56.X вместо 172.20.0.X. При попытке установить необходимый адрес выводилась следующая ошибка:

```
~/tmp/v2lab > vagrant up
==> vagrant: You have requested to enabled the experimental flag with the following features:
==> vagrant:
==> vagrant: Features: disks
==> vagrant:
==> vagrant: Please use with caution, as some of the features may not be fully
==> vagrant: functional yet.
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Clearing any previously set network interfaces...
The IP address configured for the host-only network is not within the
allowed ranges. Please update the address used to be within the allowed
ranges and run the command again.

    Address: 172.20.0.5
    Ranges: 192.168.56.0/21

Valid ranges can be modified in the /etc/vbox/networks.conf file. For
more information including valid format see:

https://www.virtualbox.org/manual/ch06.html#network\_hostonly
```

Рисунок 1: Ошибка использования адреса из запрещенного диапазона

Немного погуглив, я нашел решение:

```
~/v2lab > cat /etc/vbox/networks.conf
* 172.20.0.0/24
```

Рисунок 2: Модифицированный /etc/vbox/networks.conf

После которого появилась новая ошибка:

```

~/tmp/v2lab > vagrant up
==> vagrant: You have requested to enable the experimental flag with the following features:
==> vagrant:
==> vagrant: Features: disks
==> vagrant:
==> vagrant: Please use with caution, as some of the features may not be fully
==> vagrant: functional yet.
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Clearing any previously set network interfaces...
There was an error while executing 'VBoxManage', a CLI used by Vagrant
for controlling VirtualBox. The command and stderr is shown below.

Command: ["hostonlyif", "ipconfig", "vboxnet3", "--ip", "172.20.0.1", "--netmask", "24"]

Stderr: VBoxManage: error: Code E_ACCESSDENIED (0x80070005) - Access denied (extended info not available)
VBoxManage: error: Context: "EnableStaticIPConfig(Bstr(pszIp).raw(), Bstr(pszNetmask).raw())" at line 252 of file VB
oxManageHostonly.cpp

```

Рисунок 3: Ошибка VBoxManage E\_ACCESSDENIED

При поиске решения для этой ошибки я нашел (ссылки встроены):

- [stackoverflow](#)
- [тикет на virtualbox.org](#)

На данных ресурсах указано два способа решения, один из которых уже был применен (модификация `/etc/vbox/networks.conf`), а другой заключался в использовании VirtualBox 6.1.26 или более ранней версии (У меня стоял VirtualBox 7.X).

Однако, при установке VirtualBox 6.1 (пакет из AUR) было обнаружено, что для его работы требуются старые модули ядра (`linuxX-virtualbox-host-modules`). При этом просто старые ядра не работали, необходимы были именно старые модули (проверял 419, 515, 61), которых в свободном доступе не было! (в отличие от того же Arch; Модули ядра Arch'а не подошли).

Соответственно из этой ситуации я вижу три выхода:

- использовать виртуалку в виртуалке (внешняя 7.X, внутренняя 6.1);
- использовать Windows 10, которая стоит у меня второй системой;
- установить на рабочую машину совместимую с VirtualBox 6.1 систему (третьей).

Первый вариант отменяется ввиду возможных сложностей с настройкой (надо разрешить подобного рода развертывание) и сильных проблем с производительностью. Второй вариант тоже может быть проблемным из-за особенностей ОС Windows. Третий вариант выглядит трудноразрешимым, однако руководству-

ясь опытом установки различных дистрибутивов я посчитал его самым простым для меня. Поэтому был установлен xfce4 Debian 12:

```
~ > neofetch
_,met$$$$$gg.
,g$$$$$$$$$$$$$P.
,g$P"      ""Y$$.
,$$P'      `$$$.
,$$P      ,ggs.  $$b:
`d$$'      ,P"'   $$$
$$P        d$'    $$P
$$:        $$-    ,d$$'
$$;        Y$b._  _dP'
Y$$.      `."Y$$$$P"
`$$b      "-._
`Y$$
`Y$$.
`$$b.
`Y$b.
`"Y$b._
`""

k0tran@epsilon
-----
OS: Debian GNU/Linux 12 (bookworm) x86_64
Host: 81YQ IdeaPad 5 15ARE05
Kernel: 6.1.0-10-amd64
Uptime: 35 mins
Packages: 2253 (dpkg), 6 (snap)
Shell: zsh 5.9
Resolution: 1920x1080
DE: Xfce 4.18
WM: Xfwm4
WM Theme: Default
Theme: Xfce [GTK2]
Icons: Tango [GTK2]
Terminal: xfce4-terminal
Terminal Font: Monospace 12
CPU: AMD Ryzen 5 4500U with Radeon Graphics (6) @
GPU: AMD ATI 03:00.0 Renoir
Memory: 2454MiB / 7298MiB
```




Рисунок 4: Запуск neofetch

Установленные версии VirtualBox и Vagrant:

```
~ > VBoxManage --version
6.1.46r158378

~ > vagrant --version
Vagrant 2.3.4
```

Рисунок 5: Версии VirtualBox и Vagrant

Однако позже я узнал что скачал слишком старую версию 6.1 (небходима не позднее 6.1.26). А эта версия в свою очередь не поддерживала Debian 12, поэтому был установлен Debian 11 и установлена «правильная» версия VirtualBox:

```
> neofetch
_,met$$$$$gg.
,g$$$$$$$$$$$$$P.
,g$$$P"      ""Y$$$.
,$$P'         $$$
,$$P'         ggs.  $$$:
d$$$'         ,P"   $$$
$$P'         d$'   $$P
$$:         $$    - ,d$$'
$$;         Y$b._  d$P'
Y$$         ' "Y$$$$$P"
`$$b        "-. _
`Y$$
`Y$$
`$$b.
`Y$$b.
`"Y$b. _
  ""

k0tran@epsilon
-----
OS: Debian GNU/Linux 11 (bullseye) x86_64
Host: 81YQ IdeaPad 5 15ARE05
Kernel: 5.10.0-8-amd64
Uptime: 1 hour, 25 mins
Packages: 2515 (dpkg), 10 (snap)
Shell: bash 5.1.4
Resolution: 1920x1080
DE: Xfce 4.16
WM: Xfwm4
WM Theme: Default
Theme: Xfce [GTK2], Adwaita [GTK3]
Icons: Tango [GTK2], Adwaita [GTK3]
Terminal: xfce4-terminal
Terminal Font: Monospace 12
CPU: AMD Ryzen 5 4500U with Radeon Graphics (6) @ 2.375GHz
GPU: AMD ATI 03:00.0 Renoir
Memory: 1473MiB / 7377MiB

[Color calibration bar with 11 squares: black, red, green, blue, yellow, cyan, magenta, white, and three grayscale steps]

> vagrant --version
Vagrant 2.2.14

> VBoxManage --version
6.1.26r145957
```

Рисунок 6: Debian 11

# 1. Vagrant disk

## 1.1. Подготовительные работы

Инициализация виртуальной машины:

```
~/v2lab > vagrant init ubuntu/jammy64  
A `Vagrantfile` has been placed in this directory. You are now  
ready to `vagrant up` your first virtual environment! Please read  
the comments in the Vagrantfile as well as documentation on  
`vagrantup.com` for more information on using Vagrant.
```

Рисунок 7: Инициализация виртуальная машины

Включение функции управления дисками:

```
~/v2lab > export VAGRANT_EXPERIMENTAL="disks"
```

Рисунок 8: Включение функции управления дисками

Так же отключаем проверку обновлений. Для этого нужно раскомментировать следующую строку:

```
17 # Disable automatic box update checking. If you disable this, then  
18 # boxes will only be checked for updates when the user runs  
19 # `vagrant box outdated`. This is not recommended.  
20 config.vm.box_check_update = false  
21
```

Рисунок 9: Настройка отключения обновлений

## 1.2. Работа с дисками

Для начала имеем следующую структуру:

```
vagrant@ubuntu-jammy:~$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0        7:0      0   63.5M  1 loop /snap/core20/2015
loop1        7:1      0  111.9M  1 loop /snap/lxd/24322
loop2        7:2      0   40.8M  1 loop /snap/snapd/20092
sda          8:0      0    40G   0 disk
└─sda1       8:1      0    40G   0 part /
sdb          8:16     0    10M   0 disk
```

Рисунок 10: Дисковое пространство виртуальной машины

Здесь видно, что основной диск виртуальной машины (sda1) занимает 40 гигабайт. Для его увеличения на 20 гигабайт необходимо установить новый раздел диска 60 гигабайт. Так же добавляем подключение дополнительного жесткого диска размером 10 гигабайт (с именем extra):

```
22 # Configure disks
23 config.vm.disk :disk, size: "60GB", primary: true
24 config.vm.disk :disk, size: "10GB", name: "extra"
25
```

Рисунок 11: Конфигурация дисков виртуальной машины

Далее перезагружаем виртуальную машину и видим следующий результат:

```
vagrant@ubuntu-jammy:~$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0        7:0      0   63.5M  1 loop /snap/core20/2015
loop1        7:1      0   40.8M  1 loop /snap/snapd/20092
loop2        7:2      0  111.9M  1 loop /snap/lxd/24322
sda          8:0      0    60G   0 disk
└─sda1       8:1      0    60G   0 part /
sdb          8:16     0    10M   0 disk
sdc          8:32     0    10G   0 disk
```

Рисунок 12: Дисковое пространство виртуальной машины после изменений

На рисунке выше видно, что основной диск (sda1) теперь занимает 60 гигабайт, а так же появился дополнительный диск на 10 гигабайт (sdc)



## 2. Vagrant network

### 2.1. Vagrantfile

Приватная сеть со статичным ip адресом из диапазона 172.20.0.0/24 (был выбран 172.20.0.5):

```
37 # Create a private network, which allows host-only access to the machine
38 # using a specific IP.
39 config.vm.network "private_network", ip: "172.20.0.5", netmask: "24"
40
```

Рисунок 13: Настройка приватной сети

Публичная сеть:

```
41 # Create a public network, which generally matched to bridged network.
42 # Bridged networks make the machine appear as another physical device on
43 # your network.
44 config.vm.network "public_network"
45
```

Рисунок 14: Настройка публичной сети

Сетевое имя хоста:

```
26 # Hostname
27 config.vm.hostname = "vm1.local"
28
```

Рисунок 15: Задание сетевого имени хоста

Пробрасывание 22-го порта виртуальной машины на 3333 порт хоста:

```
29 # Create a forwarded port mapping which allows access to a specific port
30 # within the machine from a port on the host machine. In the example below,
31 # accessing "localhost:8080" will access port 80 on the guest machine.
32 # NOTE: This will enable public access to the opened port
33 config.vm.network "forwarded_port", guest: 22, host: 3333
34
```

Рисунок 16: Пробрасывание

## 2.2. Проверка всех измененных параметров

Проверяем что имя виртуальной машины поменялось на vm1 и была добавлена запись в /etc/hosts:

```
vagrant@vm1:~$ cat /etc/hostname
vm1
vagrant@vm1:~$ cat /etc/hosts
127.0.0.1          localhost

# The following lines are desirable for IPv6 capable hosts
::1              ip6-localhost      ip6-loopback
fe00::0          ip6-localnet
ff00::0          ip6-mcastprefix
ff02::1          ip6-allnodes
ff02::2          ip6-allrouters
ff02::3          ip6-allhosts
127.0.1.1        ubuntu-jammy       ubuntu-jammy

127.0.2.1 vm1.local vm1
```

Рисунок 17: Демонстрация настроек имени хоста

Рассмотрим адаптеры виртуальной машины:

```
vagrant@vm1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:f3:b3:82:d8:f8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 85772sec preferred_lft 85772sec
    inet6 fe80::f3:b3ff:fe82:d8f8/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d5:40:03 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.5/24 brd 172.20.0.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed5:4003/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:fd:6f:60 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.209/24 metric 100 brd 192.168.31.255 scope global dynamic enp0s9
        valid_lft 42575sec preferred_lft 42575sec
    inet6 fe80::a00:27ff:fefd:6f60/64 scope link
        valid_lft forever preferred_lft forever
```

Рисунок 18: Адаптеры виртуальной машины

Здесь enp0s9 имеет публичный ip 192.168.31.209, а enp0s8 имеет статический адрес 172.20.0.5.

Проверка прокинутого порта при помощи сканера nmap:

```

vagrant@vml:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:f3:b3:82:d8:f8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 85772sec preferred_lft 85772sec
    inet6 fe80::f3:b3ff:fe82:d8f8/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d5:40:03 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.5/24 brd 172.20.0.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed5:4003/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:fd:6f:60 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.209/24 metric 100 brd 192.168.31.255 scope global dynamic enp0s9
        valid_lft 42575sec preferred_lft 42575sec
    inet6 fe80::a00:27ff:fed:6f60/64 scope link
        valid_lft forever preferred_lft forever

```

Рисунок 19: Результаты сканирования nmap

## 3. Vagrant provision

### 3.1. Установка

Для начала выделим скрипт установки (из прошлой лабораторной) в отдельный скрипт:

```
4 $docker_install = <<-SCRIPT
5 # Update and install requirments
6 sudo apt-get update
7 sudo apt-get install -y ca-certificates curl gnupg
8 sudo install -m 0755 -d /etc/apt/keyrings
9 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
10 sudo chmod a+r /etc/apt/keyrings/docker.gpg
11
12 # Add the repository to Apt sources:
13 echo \
14 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
15 " $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
16 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
17 sudo apt-get update
18
19 # Install docker-engine
20 sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
21 SCRIPT
22
```

Рисунок 20: Скрипт установки docker engine

Затем необходимо добавить этот скрипт на этап provision:

```
89 # Enable provisioning with a shell script. Additional provisioners such as
90 # Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
91 # documentation for more information about their specific syntax and use.
92 config.vm.provision "shell", inline: $docker_install
```

Рисунок 21: Сам скрипт

### 3.2. Подтверждение установки

```
vagrant@vm1:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Рисунок 22: Запуск `docker run hello-world`

## 4. Vagrant multi-machine

В первую очередь скрипты для provision были выделены в отдельные переменные:

```
23 $update = <<-SCRIPT
24 sudo apt-get update
25 sudo apt-get upgrade -y
26 SCRIPT
27
28 $nginx = <<-SCRIPT
29 sudo apt-get update
30 sudo apt-get install -y nginx
31 sudo systemctl start nginx
32 SCRIPT
33
34 $add_adam = <<-SCRIPT
35 sudo apt-get update
36 sudo apt-get install -y zsh
37
38 sudo adduser adam --home /home/adam --shell /bin/zsh --disabled-password --gecos "Adam"
39 echo "adam ALL=(ALL) NOPASSWD: ALL" | sudo tee -a /etc/sudoers
40 SCRIPT
41
```

Рисунок 23: Скрипты

Затем используем их при конфигурации трех виртуальных машин:

```
47 config.vm.define "vm1" do |vm1|
48   vm1.vm.network "forwarded_port", guest: 88, host: 8080
49   vm1.vm.provision "shell", inline: $update
50   vm1.vm.provision "shell", inline: $nginx
51   vm1.vm.network "private_network", ip: "172.20.0.5", netmask: "24"
52 end
53
54 config.vm.define "vm2" do |vm2|
55   vm2.vm.disk :disk, size: "60GB", primary: true
56   vm2.vm.network "public_network"
57   vm2.vm.provision "shell", inline: $update
58   vm2.vm.provision "shell", inline: $docker
59 end
60
61 config.vm.define "vm3" do |vm3|
62   vm3.vm.network "private_network", ip: "172.20.0.6", netmask: "24"
63   vm3.vm.provision "shell", inline: $add_adam, run: "once"
64 end
```

Рисунок 24: Конфигурация vm1, vm2 и vm3

Стоит отметить, что в задании проверка функционала не заявлена обязательной. Однако для себя я запустил пару команд и, например, убедился что пользователь присутствует в системе (при помощи `id`). Так же за него можно залогиниться `sudo login adam`, предварительно установив пароль `sudo passwd adam`